Final Report— Towards Explainable, Fine-Grain Online Sexism Classification

Axel Bogos
Université de Montréal
axel.bogos@umontreal.ca

Jie Bao
Université de Montréal
jie.bao@umontreal.ca

Abstract

In this technical report, we present a comprehensive study on the identification and classification of sexist language on online social platforms, conducted in the context of SemEval 2023 Task 10. This task is particularly relevant in the current context of growing concerns about hateful speech on certain social media platforms. The task organizers provide a data set of 20,000 labeled entries from Gab and Reddit, which is divided into three hierarchical classification tasks. We employ various machine learning models, including Gaussian Naive Bayes, bi-LSTM, and Transformer-based models, to perform classification tasks and explore a variety of hierarchical classification methodologies. Our best results for each task are obtained using the RoBERTa-large model and are ranked within the top 10% of entries on the competition leaderboard for subtask B. We then outline future work, which includes better leveraging the hierarchical taxonomy in the classification architecture and enhancing explainability in a generative setting.

1 Introduction [1] [2]

1.1 Background and Motivation

The issue of online sexism is increasing, negatively impacting women, creating hostile online spaces, and perpetuating inequalities. Current automated tools designed to detect sexist content often provide limited insight, categorizing content into broad categories without further detail. As part of SemEval 2023 Task 10 (Kirk et al., 2023), this project aims to address the unique semantic challenges in the detection of sexism, such as ambivalent sexist expressions, by developing a fine-grain classification methodology that improves interpretability and explainability. In addition, an interactive demo platform will be provided to showcase the project's achievements in classification and explainability for sexist content detection. A variety of general hierarchical text classification strategies have been proposed that take advantage of different taxonomic structures, such as parent-node conditioning (Rojas et al., 2020) as well as general methodologies in hateful speech classification using model architectures such as LSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019) have been proposed; we evaluate several of these methods and models in the context of sexism detection.

1.2 Our Contributions

We have achieved the following core objectives: conducting initial data exploration, cleaning, and understanding; creating a flexible data processing pipeline adaptable to various model requirements, such as different preprocessing and embeddings for Gaussian Naive Bayes/bi-LSTM and Transformers models; setting baseline F1 scores for each task; designing a flexible training pipeline with drop-in capabilities for different models or their data-modules; establishing a flat classification baseline for a bi-LSTM model and multiple Transformer models on all tasks, developing and experimenting with multiple hierarchical classification strategies described in the literature in combination with data augmentation, and finally providing an interactive explainability demo in the form of a simple web app. Our experimental results provide a good overview of methodologies in hierarchical text classification, specifically in the context of hateful and sexist speech detection.

2 Related Work

This section outlines previous work in the detection of hate speech and sexism. Early efforts date back to 2012, when (Xiang et al., 2012) used logistic regression for the detection of offensive tweets. Recently, Transformer-based models have shown

[1] Repository
[2] WandB

high performance in various data sets related to the detection of hate speech and sexism.

Table 1: Related Work F1 Score

Model (Publication)	**2 class**	**5 class**	**23 class**
BERT + MultiCNN (Kalra and Zubiaga, 2021)	0.76	—	—
BERT + MultiCNN + data augmentation (Kalra and Zubiaga, 2021)	—	0.519	—
RoBERTa (Abburi et al., 2021)	**0.766**	—	—
RoBERTa + external knowledge (Abburi et al., 2021)	—	**0.635**	—
LaBSE (Gersome et al., 2022)	0.733	—	—
bi-LSTM + BERT (Abburi et al., 2020)	—	—	**0.583**

Model ensembling combines deep learning models, leveraging their strengths for improved performance. (Kalra and Zubiaga, 2021) pioneered the combination of BERT and CNNs for sexism classification. As well as introducing data augmentation techniques such as synonym replacement, random insertion, random swap, and random deletion, to increase the training set 8-fold.

RoBERTa, an improvement on BERT, is pretrained on more diverse datasets and longer training sequences. (Abburi et al., 2021) utilized RoBERTa and linguistic features to improve the F1 score for the 5-class classification of tweets and Gab posts, which is also used in our work (Section 3.5).

Limited research has been conducted on fine-grained multilabel sexism classification. (Abburi et al., 2020) combined bi-LSTM and attention with domain-adapted BERT, considering more categories than our study. To our knowledge, it is the first study to consider as many as 23 classes. We present a summary of all related work per number of classes considered in Table 1.

Regarding Hierarchical Text Classification (HTC), the overall task is generally perceived as harder than flat classification, since we need to consider the taxonomic relationships between each level (Liu et al., 2019a). Multiple methodologies have been proposed and explored to model these taxonomic relationships (Silla and Freitas, 2011), we go over them as part of our methods in Section 3.2.

3 Methods

3.1 Text Preprocessing

The text preprocessing class developed is a flexible solution for text preprocessing in natural language processing tasks with Spacy (Honnibal and Montani, 2017). It offers various options like lowercasing, removing spaces, punctuation, Twitter handles, URLs, stop words, lemmatizing, and tokenizing. The class sets internal states based on the specified mode (which is in turn determined based on model type, as shown in Table 5 and provides methods to preprocess individual strings or pandas Series objects.

3.2 Hierarchical Classification Strategies

3.2.1 Per-Level Classification

Per-Level classification consists of simply training multiple flat classifiers, one per level. Based on our taxonomic hierarchy, we therefore have three classifiers: a classifier for level 1 of the taxonomy with target classes {Sexist, Not sexist} which corresponds to subtask A, a classifier for level 2 of the taxonomy with target classes {Threats, Derogation, Animosity, Prejudiced Discussion} which corresponds to subtask B and a classifier for the whole third level of the taxonomy with target classes {Threats of harm, . . . , Supporting systemic discrimination against women}. Notice that this paradigm does not take into account the inter-level relationship and the taxonomic constraints between classes. A single example X can thus be predicted to be of classes a_2, b_1, c_5 for subtasks A, B and C where b_1 is not a child class of a_1, and c_5 is not a child class of b_1.

3.2.2 Per-Level Classification + Exhaustive Beam-Search

Although beam search is typically used in the context of neural machine translation (Freitag and Al-Onaizan, 2017), by viewing the hierarchical labels at each level of the HTC taxonomy as a sequence of labels, we can apply the beam search algorithm as a means of level-by-level decoding (Rojas et al., 2020). This method presents an advantage over the per-level classification strategy presented in Section 3.2.1, as it favors the model to jointly optimize

the probability of a label sequence across the entire taxonomy. At each decoding step i, (in our case, we have a total of 3 decoding steps given 3 hierarchical levels), we consider the probability of $P(y_k^i|x, y^1, y^2, \dots y^{i-1})$ where y^i is a class probability output of a per-level classifier and k is our beam size. But since, unlike neural machine translation vocabulary, we have a rather limited number of classes to choose from at each decoding step, we may actually exhaustively consider all joint probabilities at each decoding level, effectively matching our beam size k with the number of classes at each level. Thus, we wish to leverage the taxonomic relationships between each level and increase the bias towards consistent label sequences by using beam-search in combination with the softmax output of per-level classifiers.

3.2.3 Per-Parent Classification

This method, also referred to as Parent-Node Conditioning (Rojas et al., 2020), consists of training P classifiers, where P is the number of nonleaf nodes in our taxonomy. Then, every example X is classified from the root classifier and propagated throughout the taxonomy as the highest likelihood class of each parent node at each level until a leaf node is reached. For example, an example classified as *Sexist* by the first parent classifier may be classified as any of the child classes of the *Sexist* class. If it is classified as *Threat* by the next parent classifier, it can then only be classified as a child class of *Threat*, which are *Threat of harm or Incitement and encouragement of harm* class, both of which are leaf nodes and do not have classifiers attached to them.

3.2.4 Per-Level Classification + Hierarchical Masking

In this method, we combine some of the insights proposed with per-level classification and beam search in 3.2.2 and parent-node conditioning in 3.2.3. Here, we use per-level classifiers as with the beam search approach, but directly inject knowledge of the taxonomic constraints by masking non-child classes based on the previous highest likelihood classification. For example, we start at the first level without prior and obtain the predicted class at level 1 by simply maximizing the class probabilities. At the subsequent level, we condition our per-level classifier on this previous label sequence by masking the class probabilities of the current-level classifier to 0. This combined approach allows to sidestep issues arising from parent-node conditioning such as a very small subset of the data being available to lower-level parent classifiers while still providing consistent label sequences and injecting taxonomic knowledge in the classification strategy.

3.3 Gaussian Naive Bayes

The Gaussian Naive Bayes models are taken from the scikit-learn (Pedregosa et al., 2018) implementation, specifically the MultinomialGB module. The text is pre-processed with all internal flags described in Section 3.1 activated. Then, a scikit-learn CountVectorizer is used to encode the text. Finally, models are fitted on the training set and evaluated on the validation set and test set (we include the validation set as that is the publicly available leaderboard scores for the Sem-Eval task). No particular run arguments are needed, nor any parameter optimization done. We report our results in Table 7.

3.4 Bi-LSTM

The bi-LSTM (Graves and Schmidhuber, 2005)(Hochreiter and Schmidhuber, 1997) module is developed as a Pytorch-Lightning module (Falcon and The PyTorch Lightning team, 2019). The architecture is customizable via run arguments, but the default configuration uses the full preprocessing pipeline, has a 300 unit embedding layer and 2 bidirectional layers of 256 hidden units, and a dropout layer before a final classification head. The default optimizer is AdamW (Loshchilov and Hutter, 2019) with a learning rate of 0.001. We train a distinct bi-LSTM for each task and report our results in Table 7. An overview of the training arguments can be found in Table 5. Although no formal hyperparameter optimization was performed for bi-LSTM, we did try a few different configurations of embedding dimensions, hidden dimensions, and number of layers.

3.5 Transformer Models

We explore a number of Transformer-based (Vaswani et al., 2017) models such as BERT (Devlin et al., 2019), RoBERTa(Liu et al., 2019b), distilBERT and distilRoBERTa (Sanh et al., 2019). To easily iterate between models, we developed a custom abstract Transformer Pytorch Lightning module that instantiates any of these models (or their size variations) as the backbone of the classification module with an additional classification head

consisting of a hidden fully connected layer of 768 units, a dropout layer, and a final fully connected classification layer. This paradigm of feature extractor and classification head is re-used across all the hierarchical classification methodologies discussed in Section 3.2, albeit with a different number of classification heads and different handling of each heads depending on the approach. All transformer models are loaded as pretrained from HuggingFace (Wolf et al., 2020), as well as their respective tokenizers. In all cases, the feature extractor is frozen. The latter is provided to the custom lightning datamodule (more details are provided in Section 4.1. The default run arguments, such as optimizer, warm-up steps, scheduler, and learning rate, can be found in Table 6. For Per-Parent classification, due to the high number of classifiers, we use a single, shared feature extractor based on the model of choice, and then distinct classification heads for each parent classifier.

3.6 Data Augmentation

The class imbalance in the data set can be visualized in Figure 3. We observe that for tasks B and C, the dataset is significantly reduced down to 30% of its original size, as seen in Figures 4 and 5. Data augmentation is a common technique to increase the size of a smaller data set and can be applied in the context of text classification to improve performance (Wei and Zou, 2019). Five simple yet powerful techniques that we used are back-translation (MarianMT), synonym replacement, random insertion, random swap, and random deletion. We used distilBERT to compare the performance of each technique and found random insertion and synonym replacement performed the most effectively for our problem. In the next sections, we explain our approach to further augment the data set.

3.6.1 Contextual Word Embedding Augmentation

Following the success of the random insertion and synonym replacement method. We revisited our approach to obtain a better representation of the method. The original approach consists of using the NLTK package to perform word insertion and synonym replacement. This approach was quick and provided good results with the distilBERT model. However, it lacked the contextual information to best capture the semantics of an ambivalent sexist sentence and could potentially add noise to the data set. As an alternative, we used contextual embeddings for better word insertion and synonym replacement (Ma, 2019). This method is better at capturing the meaning of the sentence by extracting the hidden state of the word which contains contextual information that the previous method did not have. From the extracted hidden state, we perform a linear transformation using the weight matrix from the pre-trained Roberta-large model which maps to the hidden state vector to the vocabulary size. The generated word from the softmax sample is then added to the previously selected position in the sentence.

3.6.2 Pseudo-labeling Strategy

As part of SemEval task 10, we are provided a an extremely large data set of unlabeled instances compared to the labeled data, S. In this project, we seek to prove the benefit and compatibility of using unlabeled data in the training process of a text classifier for ambivalent sexism detection. The semi-supervised method consists of iteratively learning a classifier by assigning pseudo-labels from the unlabeled dataset, X_u. Pseudo-labeled examples with a margin greater than a minimum accuracy margin will be removed from X_u. In practice, this process should be completed until X_u is empty. However, due to resource limitation, we have done one self-training pass for each model. From (Amini et al., 2022), we seek to minimize the following loss for the classifier h.

$$\frac{1}{m}\sum_{(x,y)\in S} l(h(x,y)) + \frac{\gamma}{|X_u|}\sum_{(x,\tilde{y})\in X_u} l(h(x,\tilde{y})) + \lambda||h||^2$$

In our project, we tested the approach on the distilRoBERTa and RoBERTa large models. Therefore, 6 classifiers are trained separately. A schematic figure is included in the appendix A that summarizes the process.

3.7 Explainability Demo

An important component in properly identifying and taking appropriate action with hateful content online is to provide users or content creators with clear explanations as to why certain content was labeled hateful or sexist (Vidgen et al., 2019). Therefore, we intend on providing an explainability module demonstration, showcasing model predictions given a text excerpt and a visual or textual explanation of the classification for each task. Due to time constraints, our ambitions in this section of

the project had to be slightly tempered. We provide a locally deployed Streamlit (Streamlit, 2023) web-app which allows the user to either select from some predefined text examples or to enter his own, obtain predictions from our second-best performing model (with roberta-base (Liu et al., 2019b)) and a visual explanation of the subtask A classification as a Shap text and force plots. These plots are based on the SHAP (SHapley Additive exPlanations) values associated with each token. SHAP values are a game-theoretic approach to assess the impact of each feature (in this case, of each token) on the final prediction. An example of the app output can be seen in Figure 1.

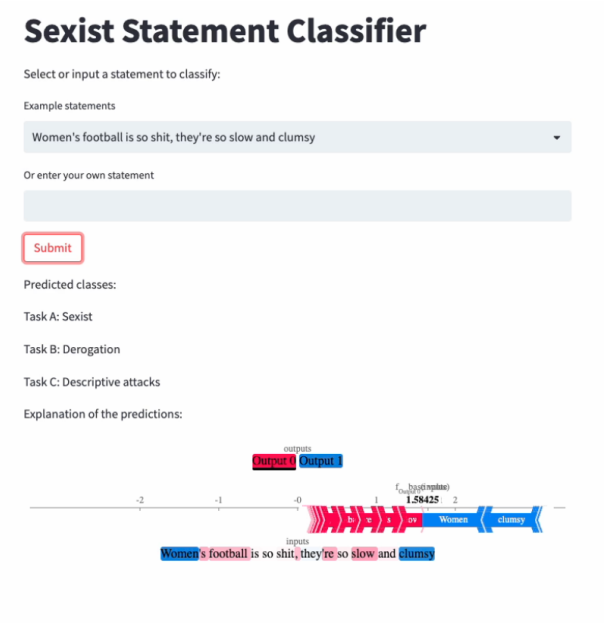

Figure 1: Explainability Web App Demo

4 Experiments

4.1 Dataset

The data set contains 20,000 labeled entries, 10,000 from Gab and 10,000 from Reddit, provided by the task organizers (Kirk et al., 2023). The task is structured into three hierarchical classification tasks, as illustrated in Figure 2. Subtask A involves a binary classification of sexist/nonsexist content, while subtask B focuses on a 4-way classification, conditional on the content being sexist. Subtask C consists of 2-way, 3-way, 4-way, and 2-way classification problems, respectively, conditional on each of the 4 subtask B classes.

All-women annotators reach decisions through majority voting. Although our project will not join the

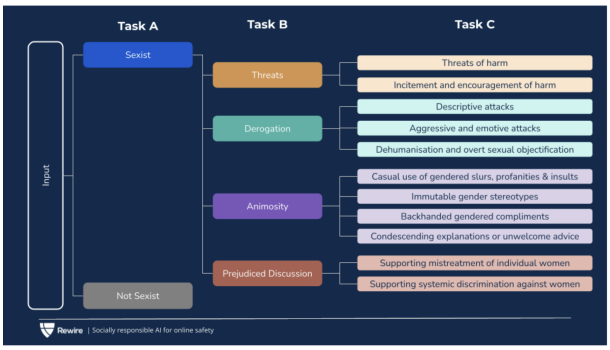

Figure 2: Hierarchical Label Schema Overview

Sem-Eval 2023 task as it is already closed, we will use its public leaderboard for benchmarking. The appendix Figure 7 illustrates the sequence lengths, revealing that the maximum length is below the 512 token limit of the BERT models. To facilitate data handling, we use Pytorch Lightning data modules to handle data loading for each task. We provide some examples of segments for subtask A in Table 2, for subtask B in Table 3 and for subtask C in Table 4.

The dataset is also highly imbalanced. As each subtask class is conditioned on parent classes in the parent subtask, this imbalance issue only grows with each additional taxonomic level. We provide an overview of the distribution of classes for each subtask; respectively, in Figure 3 for subtask A, in Figure 4 for subtask B, in Figure 5 for subtask C. This imbalance is partly handled through data augmentation, as described in 3.6.

Table 2: Data Examples Task A

Text	Label
I literally HATE the Republicans. I am one but they are a disgrace	not sexist
[USER] Leg day is easy. Hot girls who wear miniskirts get asked out.	sexist

Table 3: Data Examples Task B

Text	Label
I literally HATE the Republicans. I am one but they are a disgrace	none
[USER] Leg day is easy. Hot girls who wear miniskirts get asked out.	3. animosity

4.1.1 Augmented Dataset

As mentioned in Section 3.6.2, we trained 6 classifiers and the amount of pseudo-labels added to the

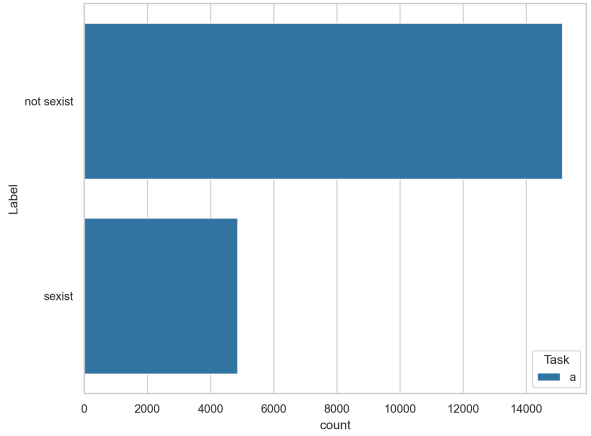

Figure 3: Label Distribution Task A

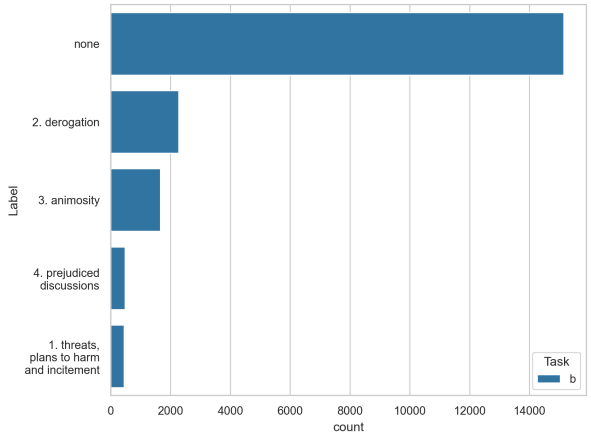

Figure 4: Label Distribution Task B

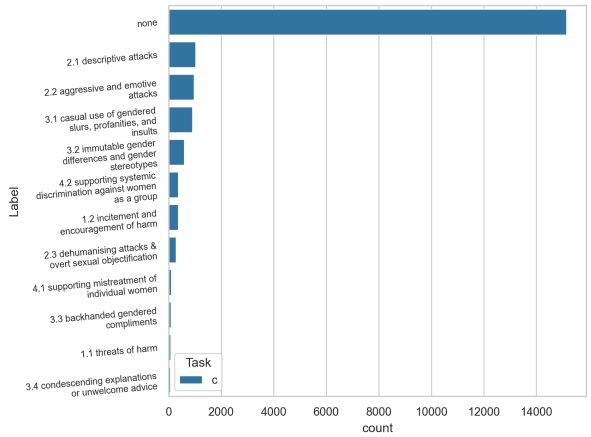

Figure 5: Label Distribution Task C

Table 4: Data Examples Task C

Text	Label
I literally HATE the Republicans. I am one but they are a disgrace	none
[USER] Leg day is easy. Hot girls who wear miniskirts get asked out.	3.3 backhanded gendered compliments

training set is different. The pseudo-labels were also highly imbalanced and were balanced manually using the context embedding method. A first pass with only the given labeled data is done. Then using the saved model stored in wandb, we classify the unlabeled dataset. The confidence margin ranged from 0.75 to 0.95 depending on the output. Instances that respect the margin are added in the labeled dataset. A summary of the augmented dataset can be visualized in the appendix A.

4.2 Baselines

Our baselines are established with a simple Gaussian naive Bayes (GNB) model, trained from scratch by us. A distinct baseline is established for each of the three tasks. We have adjusted our definition of the lower and upper baselines from the initial project proposal. We initially proposed that all models trained in a "flat classification" paradigm would be considered as part of the baseline;however, we have achieved good results with this paradigm, especially with transformer-based models, and thus we treat them as results and not as baselines. Our baseline pipeline consists of a simple text preprocessing pipeline, namely, converting strings to lowercase, removing handles, URLs, and special characters, replacing multiple whitespaces with a single space, removing punctuation, removing stop words, and performing lemmatization. The text examples are then numerically represented with sklearn *CountVectorizer*, taking care not to leak information between training and validation / test sets, and are finally used to train the GNB models. The results of these baselines can be found in Table 7. No external data is used in the training of our baselines.

4.3 Evaluation Methods

Our main evaluation metric is the macro-averaged F1 score. This matches the evaluation metric used in both this Sem-Eval task (Kirk et al., 2023)

and the aforementioned IberLEF task (Rodríguez-Sánchez et al., 2021), thus giving us the means to directly compare our results with the leaderboards of these tasks. Although we are not yet working on the explainability module of our predictive models, we expect that the evaluation of the quality of the explanations (given a correct prediction) will be qualitative and based on human evaluation rather than quantitative, as there are no general standardized explainability evaluation metrics (Danilevsky et al., 2020).

4.4 Experimental Details

Beside the Gaussian Naive Bayes, which may easily run on a standard laptop, all our experiments are run on a Tesla T4 GPU on Google Colab. To run in colab, we simply clone our code repository and execute one of the main training scripts with the appropriate run arguments. All of our experiments are logged and tracked using Weights and Biases (WandB) (Biewald, 2020) and are available here. We do not comment on GNB training, as it has been explained in Sections 3.3 and 4.2. All our experiments are executed with a main training script, which parses the run arguments, initializes logging, loads the appropriate model and data module based on run arguments, and finally creates and fits a PytorchLightning Trainer. Logging is done both on WandB and locally in a dynamically created log directory for each run. The best checkpoint based on loss is also saved as an artifact both locally and on WandB.

4.4.1 Bi-LSTM

We provide hyperparameters and run arguments of the bi-LSTM model in Table 5. We discuss the details of the implementation in Section 3.4.

4.4.2 Transformer Models

First, we proceed with the training of per-level classifiers. Per-level models classification heads are then re-used as subcomponents of wrapper modules for per-level classifier + beam search and hierarchical masking. Per-parent node conditioning requires retraining distinct classification heads for each of the sub-sub-parent-conditioned tasks, requiring a total of 6 classification heads. We provide hyperparameters and run arguments of the standard per-level Transformer model configuration in Table 6. We have discussed the details of the implementation in Section 3.5.

Table 5: Training Arguments of the Bilstm Model

Hyperparameter	Value
dropout	0.1
optimizer	AdamW
lr	0.001
preprocessing_mode	standard
max_token_length	128
embedding_dim	300
hidden_dim	256
num_layers	2
bidirectional	True
batch_size	32
max_epoch	10
early_stopping	True
patience	3

Table 6: Training Arguments of the Transformer Model

Hyperparameter	Value
model	{roberta-base, . . . }
optimizer	AdamW
lr	5e-06
step_scheduler	5
n_warmup_steps	5
preprocessing_mode	none
max_token_length	128
batch_size	16
max_epoch	10
early_stopping	True
patience	3

4.5 Results

We present an overview of our results in Table 7. All our results are measured as the macro-averaged F1 score; hence, since we are using the same test set as the original Sem-Eval task, we can directly compare our results with the competition results. Our results also compare favorably with those of related work, as shown in Table 1. We see that we achieve quite good results with per-level classification and with per-level classification + data augmentation. Methods that leverage taxonomic relationships all perform relatively well in Task A; however, their performance degrades significantly in subsequent tasks, notably for hierarchical masking and parent-node conditioning. Our results also perform well in the overall competition; we show the difference between our top score and the overall top score of the competition among 129 participants and our leaderboard ranking in Figure 6. We further ana-

Table 7: Overview of Results

Model	Task A	Task B	Task C
Per-Level Classification			
Gauss-Naive Bayes Baseline	0.6912	0.3218	0.1748
bi-LSTM	0.7342	0.2668	0.1329
distilbert-base-uncased	0.7991	0.5819	0.2794
bert-base-uncased	0.8083	0.5958	0.2827
distilroberta-base	0.8194	0.6113	0.3119
roberta-base	0.8297	0.6493	0.4035
robert-large	0.8555	0.7205	0.5122
Per-Level + Exhaustive Beam Search			
distilroberta-base	0.7077	0.3664	0.2246
Per-Level Classification + Hierarchal Masking			
distilroberta-base	0.6542	0.1125	0.0452
Per-Level Classification + Data Augmentation			
distilroberta-base	0.8239	0.6443	0.3683
robert-large	**0.8617**	**0.7219**	**0.5201**
Parent-Level Classification			
distilroberta-base	0.8194	0.6199	0.05246
Competition Results			
Top Leaderboard Score	0.8843	0.7793	0.6238
Leaderboard Ranking for our Best Result (/129)	20	13	16

lyze our results, their possible shortcomings, and comparisons with the leaderboard scores in Section 4.6.

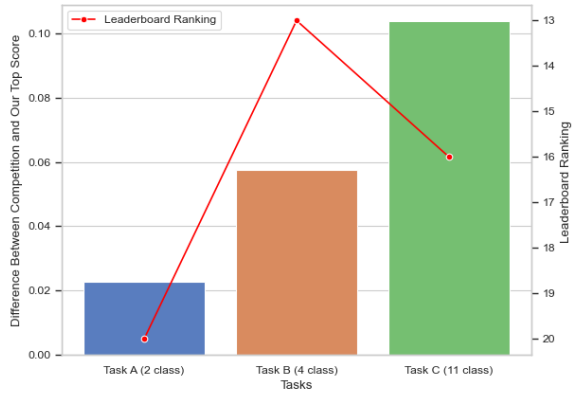

Figure 6: Difference in score between Competition Winner and our Best Model and Leaderboard Ranking of our Best Model

4.6 Analysis

A possible shortcoming and a reason for very poor performance in tasks at the lower taxonomic level, such as Task C for parent-node conditioning and hierarchical masking is the forward error propagation at each level. By providing no opportunity to the model to correct the joint-probability in lower levels in case of a misclassification in an upper level, we propagate errors exponentially through the taxonomic levels. Furthermore, by observing in Figure 6 that our results are simultaneously better ranked in task B and C in the overall competition than for task A, but also further away from the top score of the competition in the latter tasks, we may deduce that most participants struggle to leverage taxonomic relationships, and in fact fine-tuning our per-level classification and augmenting the imbalanced data set has been more fruitful for us. Although our best performance is from *Per-Lever Classification + Data Augmentation*, we have seen diminishing returns from the RoBERTa large classification. This is possibly due to the increased data size, quality, and diversity requirements from a larger model. Nevertheless, there is no doubt that better performances will be achieved if taxonomic relationships are properly incorporated.

5 Conclusion

We have shown that data augmentation techniques coupled with semi-supervised techniques can increase performance in text classification in the context of fine-grain sexism classification. Although the improvements were minor, we showed the feasibility of self-training that resulted in a better class balance.

5.1 Future Work

Our future work includes two main tracks: improving our methodologies to leverage taxonomic relationships at the architectural levels of our models and exploring model explainability in generative settings. For the first, exploring label embedding methods such as hyperbolic embedding (Chatterjee et al., 2021) is promising towards including taxonomic relationships and avoiding the aforementioned error propagation. An investigation of softer regularization and hierarchical masking may also prove fruitful. Regarding explainability, we wish to investigate using a representation of our per-level classifiers as the basis of a prompt a large-language model, say LLaMA (Touvron et al., 2023) or simply the Chat-GPT API (OpenAI, 2023), to generate an explanation for the classification, nuanced based on the actual probability distribution. This may be particularly useful given the fuzzy nature of certain class definitions and provide a more holistic and human-like understanding of what makes a statement hateful across multiple classes rather than an absolute argmax.

References

Harika Abburi, Pulkit Parikh, Niyati Chhaya, and Vasudeva Varma. 2020. Fine-grained multi-label sexism classification using semi-supervised learning. In *Web Information Systems Engineering – WISE 2020: 21st International Conference, Amsterdam, The Netherlands, October 20–24, 2020, Proceedings, Part II*, page 531–547, Berlin, Heidelberg. Springer-Verlag.

Harika Abburi, Shradha Sehgal, Himanshu Maheshwari, and Vasudeva Varma. 2021. Knowledge-based neural framework for sexism detection and classification. In *IberLEF@SEPLN*.

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040*.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Soumya Chatterjee, Ayush Maheshwari, Ganesh Ramakrishnan, and Saketha Nath Jagaralpudi. 2021. Joint learning of hyperbolic label embeddings for hierarchical multi-label classification.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable ai for natural language processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Shimi Gersome, Jerin Mahibha, and Durairaj Thenmozhi. 2022. Sexism identification in social media using deep learning models.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Amikul Kalra and Arkaitz Zubiaga. 2021. Sexism identification in tweets and gabs using deep neural networks. *CoRR*, abs/2111.03612.

Hannah Rose Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. 2023. SemEval-2023 Task 10: Explainable Detection of Online Sexism. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*. Association for Computational Linguistics.

Liqun Liu, Funan Mu, Pengyu Li, Xin Mu, Jing Tang, Xingsheng Ai, Ran Fu, Lifeng Wang, and Xing Zhou. 2019a. NeuralClassifier: An open-source neural hierarchical multi-label text classification toolkit. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–92, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

OpenAI. 2023. Gpt-4 technical report.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2018. Scikit-learn: Machine learning in python.

Francisco Rodríguez-Sánchez, Jorge Carrillo de Albornoz, Laura Plaza, Julio Gonzalo, Paolo Rosso, Miriam Comet, and Trinidad Donoso. 2021. Overview of exist 2021: sexism identification in social networks. In *Proces. del Leng. Natural*.

Kervy Rivas Rojas, Gina Bustamante, Arturo Oncevay, and Marco A. Sobrevilla Cabezudo. 2020. Efficient strategies for hierarchical text classification: External knowledge and auxiliary tasks.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Carlos N. Silla and Alex A. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1–2):31–72.

Streamlit. 2023. Streamlit.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Bertie Vidgen, Alex Harris, Dong Nguyen, Rebekah Tromble, Scott Hale, and Helen Margetts. 2019. Challenges and frontiers in abusive content detection. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 80–93, Florence, Italy. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 1980–1984, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 Additional Figures and Tables

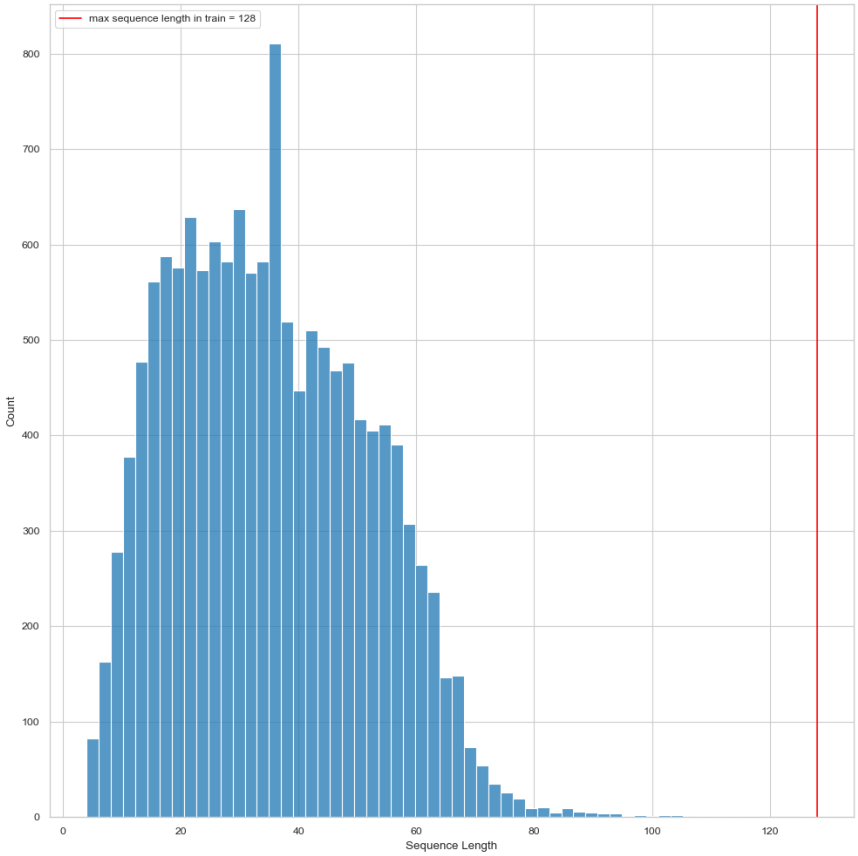

Figure 7: Data Set Sequence Length Distribution

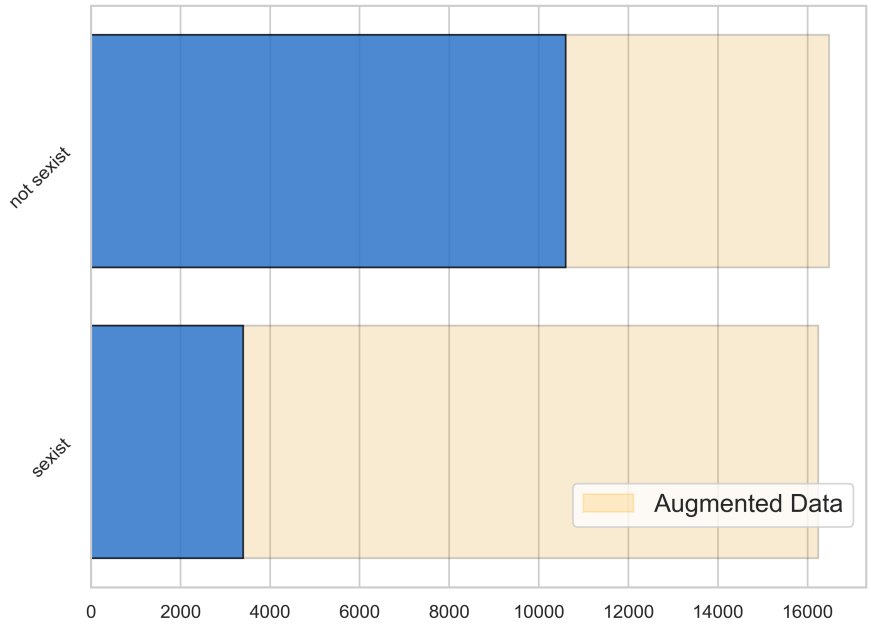

Figure 8: Data Augmentation Task A - DistilRoBERTa

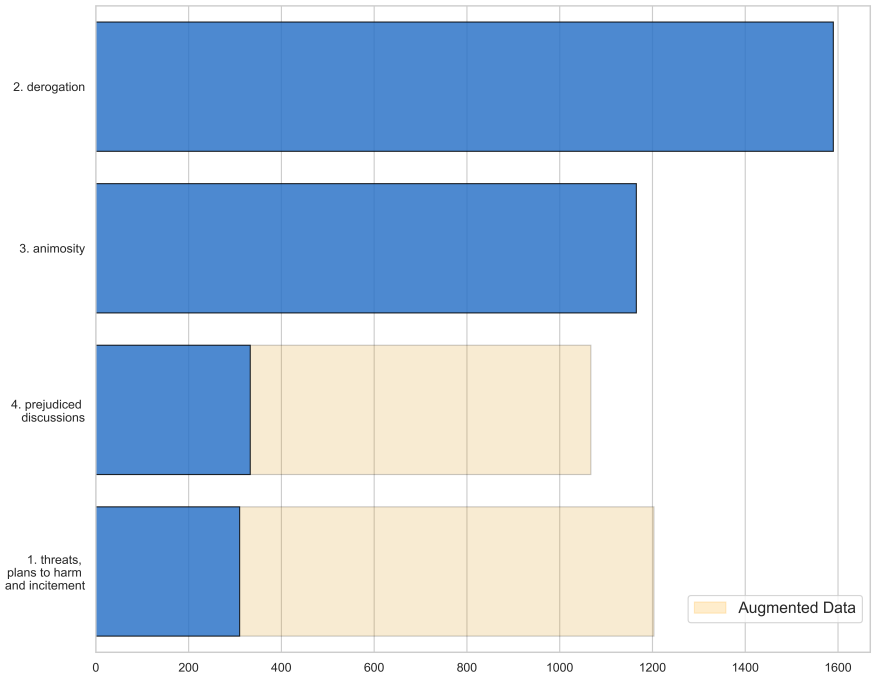

Figure 9: Data Augmentation Task B - DistilRoBERTa

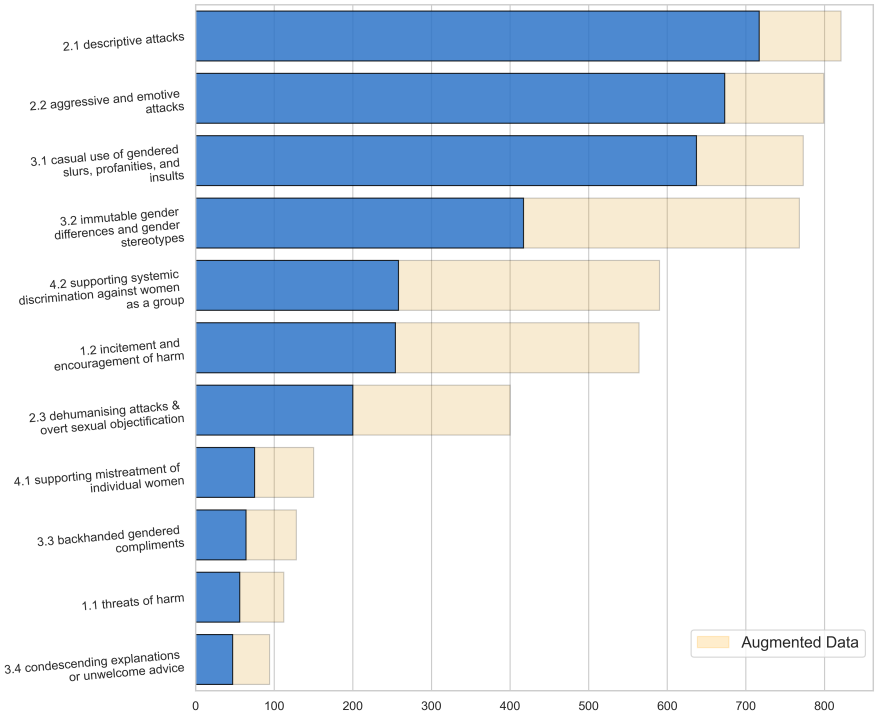

Figure 10: Data Augmentation Task C - DistilRoBERTa

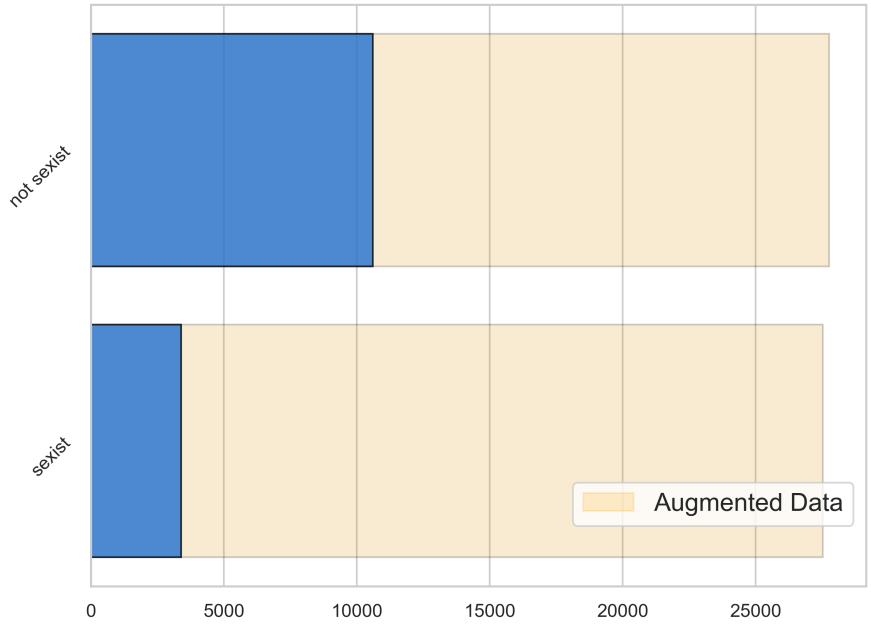

Figure 11: Data Augmentation Task A - RoBERTa-Large

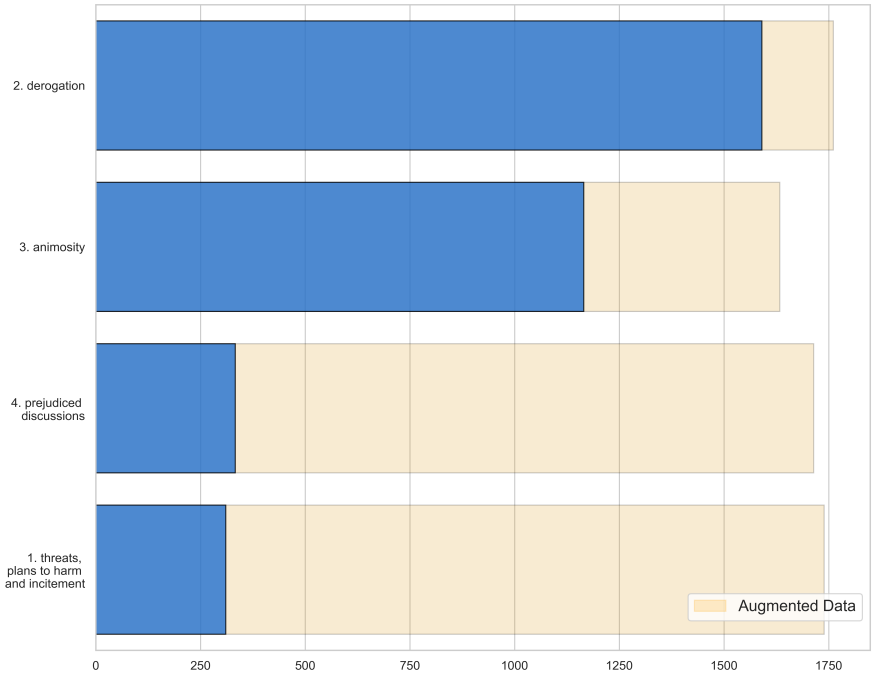

Figure 12: Data Augmentation Task B - RoBERTa-Large

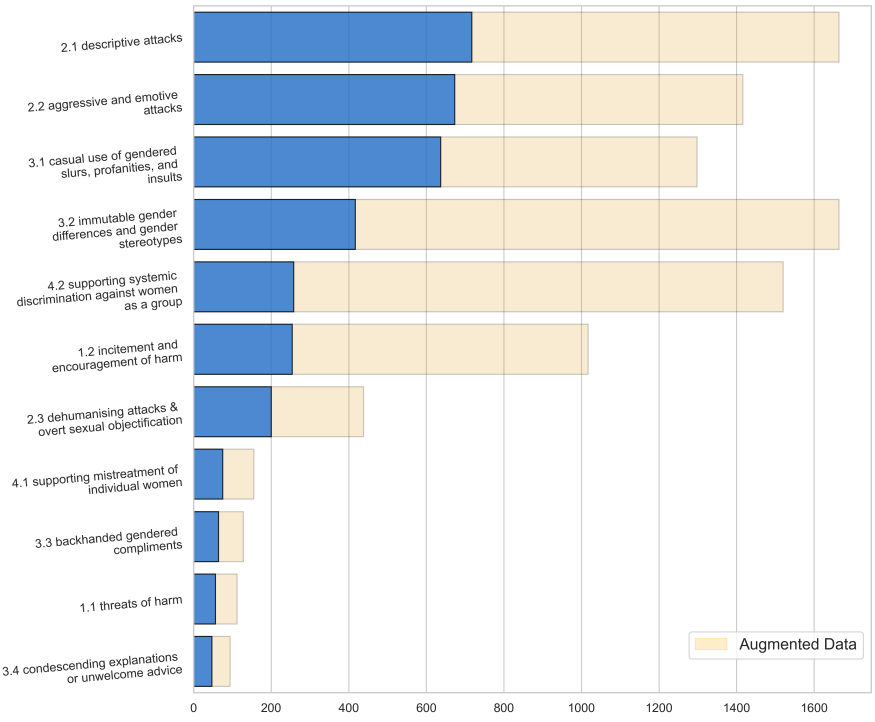

Figure 13: Data Augmentation Task C - RoBERTa-Large

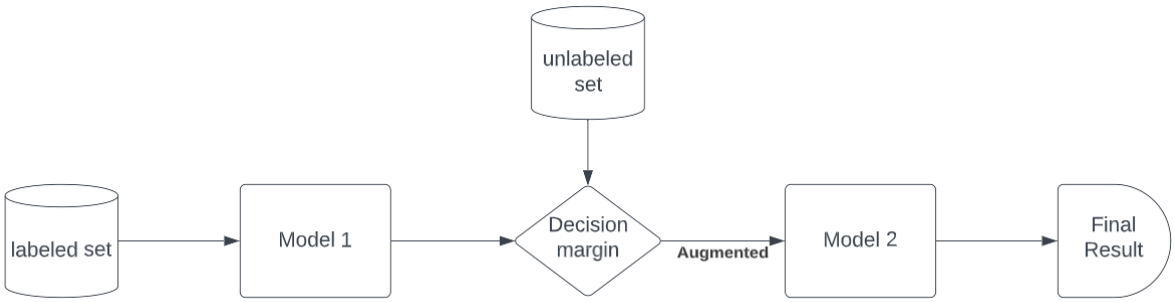

Figure 14: Self training process